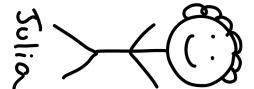


love this?
more zines at
→ wizardzines.com ←



hello!
quick note: 5 of
these tools are Linux-only



Silvia

Linux only tool

BSD/ Mac equivalent

ip	ifconfig, route
tc	dummynet (?) (BSD)
ss	netstat
iptables	pf (BSD)
ethtool	ifconfig, kind of (?)

ethtool

23

ethtool is for people who need to manage physical networks

why no internet??
oops it would help
server if your ethernet cable was plugged in

ethtool `eth0`
name of network interface
this tells you:
- is it even connected?
("link detected")
- speed
- lots more

--show-offload
--offload
your network card can do a lot for you! Like computing checksums. This is called "offloading". This lets you see / change configured offloads.

--identify INTERFACE
blink the light on the ethernet port. good if you have multiple ports! and cute

-s
change speed/duplex / other settings of an interface
ethtool -s eth0 speed 100

-i INTERFACE
show statistics like bytes sent. works for wifi interfaces too.

show firmware info

conntrack

22

conntrack

not a command line tool:

it's a Linux Kernel system
for tracking TCP / UDP
connections.

It's a kernel module
called nf_conntrack

conntrack is used for:

- NAT (in a router!)
 - firewalls (eg only allow outbound connections)
- You control it with iptables rules.

how to enable conntrack enable:

sudo modprobe nf_conntrack

check if it's enabled:

lsmod | grep conntrack

change table size with the sysctl
'net.netfilter.nf_conntrack_max'

conntrack has a table
of every connection

Each entry contains:

- src + dest IP
- src + dest ports
- the connection state
(eg TIME_WAIT)

moral: be careful about
enabling conntrack!





if the conntrack table
gets full, no new
connections can start



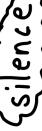


Table of contents

dig.....4	tcpdump.....10-11	ip.....18
ping.....5	tshark.....12	ss / netstat.....19
curl.....6	ngrep.....13	iptables.....20
nmap.....7	openssl.....14	tc.....21
netcat.....8	mitmproxy.....15	conntrack.....22
socat.....9	misc tools.....16	ethtool.....23
ssh.....17	

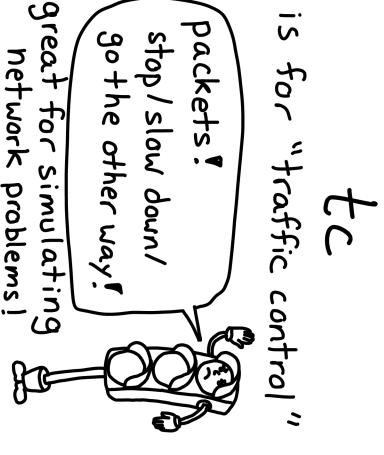
dig

dig makes DNS queries!

```
$ dig google.com
answers have 5 parts:
query: google.com
    TTL: 22
    class: IN (for "internet")
    record type: A
    record value: 172.217.13.110
```

dig +trace domain

traces how the domain gets resolved, starting at the root nameservers if you just updated DNS, dig +trace should show the new record



dig @ 8.8.8.8 domain

dig @server lets you pick which DNS server to query! Useful when your system DNS is misbehaving :)

dig -x 172.217.13.174

makes a reverse DNS query - find which domain resolves to an IP! Same as dig ptr 174.13.217.172.in-addr.arpa

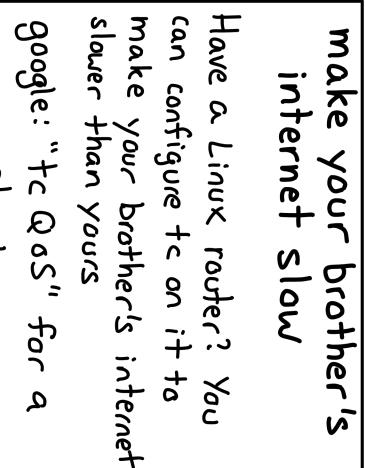
tc

make your internet slow

```
sudo tc qdisc add dev w1p3s0 root netem
delay 500ms < delay packets by 50ms
and fast again:
sudo tc qdisc del dev w1p3s0 root netem
```

netem rules

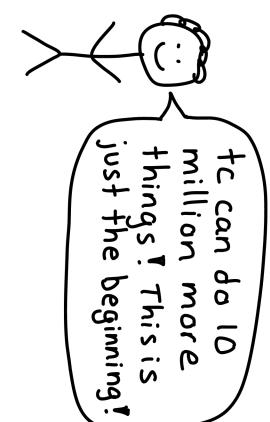
netem ("network emulator") is a part of tc that lets you:
drop, **duplicate**, **delay**, **corrupt** packets. See the man page:
\$ man netem



Have a Linux router? You can configure tc on it to make your brother's internet slower than yours
google: "tc QoS" for a start

Show current tc settings

```
tc qdisc show
tc class show dev DEV
tc filter show dev DEV
```



iptables

20

iptables lets you create rules to match network packets and accept/drop/modify them. It's used for firewalls and NAT.

tables have chains
chains have rules
tables: filter, nat, mangle, raw, security
chains: INPUT, FORWARD, etc, PREROUTING, etc
rules: like -s 10.0.0.0/8 -j DROP

-j TARGET

Every iptables rule has a target (what to do with matching packets). Options:
→ ACCEPT / DROP / RETURN
→ the name of an iptables chain (e.g.
an extension (man iptables-extensions))
Popular: DNAT, LOG, MASQUERADE

tables have different chains
filter: INPUT/OUTPUT/FORWARD
mangle: INPUT/OUTPUT/FORWARD/
PREROUTING/POSTROUTING
nat: OUTPUT/PREROUTING/POSTROUTING
It helps to learn when
packets get processed by a
given table/chain (e.g.
filter+Output = all locally
generated packets)

iptables-save
This prints out all
iptables rules. You can
restore them with
iptables-restore
but it's also the easiest
way to view all rules!

you can match lots
of packet attributes
-s: src ip -p: tcp/udp
-d: dst ip -i: network
interface
-m: lots of things!
(bpf rules! groups! ICMP type!
cpu! conntrack state! more!)
For more, run:
\$ man iptables-extensions

Ping & traceroute

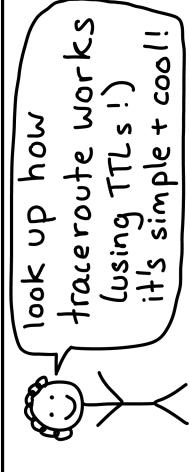
5

ping checks if you
can reach a host and
its latency
\$ ping health.gov.au
output:
.... time=253ms...
Australia is 17,000 km from me.
at the speed of light it's still far!

myth: if a host doesn't
reply to ping, that
means it's down
Some hosts never respond
to ICMP packets. This is why
traceroute shows ... sometimes.

ping 

traceroute tells you
the path a packet takes
to get to a destination


mtr
like traceroute, but
nicer output! try it!


example traceroute
\$ traceroute health.gov.au
1: 192.168.1.1 3ms ← router
2: ...yul.ebox.ca 12 ms → ISP
crossing the US!
8: NYC4.ALTER.NET 24 ms
9: SAC1.ALTER.NET 97 ms
16: health.gov.au 253ms →
here the packet crossed the USA!
from NYC → Sacramento!

CURL

curl
it's my favourite way to make HTTP requests!
great for testing APIs!
[\\$ curl wizardzines.com](http://curl.wizardzines.com)

-i	show response headers
-I	show <u>only</u> response headers (makes a HEAD request)
-X POST	send a POST request instead of a GET (-X PUT etc works too)

-V	show request headers & more
-k	insecure: don't verify SSL certificates
--connect-to ::IP	send request to IP instead. or hostname use before changing DNS to a new IP

★ copy as curl ★	Have something in your browser you want to download from the command line? In Firefox / Chrome / Safari: Developer Tools → Network tab → right click on the request → copy as curl (can have sensitive info in cookies!)
------------------	--

SS

19

① Use ss to find the process ID using the port	I can't start my server because it says something is using port 8080!
② Kill the other process!	"socket statistics"

★ tuna, please! ★	\$ ss -tunap1 `here the 'a' doesn't do anything use ss! It shows all the running servers
-------------------	--

-n	use numeric ports (80 not http)
-P	show PIDs using the socket
TONS of information	-i -m -o

netstat	netstat -tunap1 and ss -tunap1 do the same thing
-t : listening	netstat is older and more complicated. If you're learning now I'd recommend ss!

listening or connections?	which protocols?
non-listening/ established	default: all -t : TCP -u : UDP -X : unix domain sockets

IP

18

IP

Linux only
lets you view + change network configuration.
ip OBJECT COMMAND
→
addr, link add, show, delete, etc
neigh, etc
Here are some ways to use it!

ip addr list

shows ip addresses of your devices. Look for something like this:
2: eth0:
link/ether 3c:97...
inet 192.168.1.170/24

change your MAC address

good for cafes with time limits 😊

```
$ ip link set wlan0 down  
$ ip link set eth0 address  
3c:a9:f4:d1:00:32  
$ ip link set wlan0 up  
$ service network-manager restart ← or whatever you use
```

ip link

network devices! (like eth0)

ip neigh

view/edit the ARP table

ip xfrm

is for IPsec

ip route list

displays the route table.
↳ my router
default via 192.168.1.1
169.240.0.0/16 dev docker0
...

to see all route tables:
ip route list table all

ip route get IP

what route will packets with \$IP take?

-- color

pretty colourful output!

-- brief

show a summary

nmap

7

nmap lets you explore a network

which ports are open?
What hosts are up?

security people use it a lot!

find which hosts are up

```
$ nmap -sn 192.168.1.0/24
```

↑
my home network

-sn means "ping scan".
(not -s + -n, it's -sn)
just finds hosts by pinging every one,
doesn't port scan

fast port scan

```
$ nmap -ss -F 192.168.1.0/24  
just sends a SYN packet to check if each port is open.
```

I found out which ports my printer has open!
HTTP 80
HTTPS 443
Printer 515
IPP 631
JetDirect 9100

-F

scan less ports: just the most common ones

-T4 or -T5

scan faster by timing out more quickly

aggressive scan

↑
nmap -v -A scanme.nmap.org part, server version, even OS

-Pn
skip doing a ping scan and assume every host is up.
good if hosts block ping (lots)

check TLS version ↴ and ciphers

check if your server still supports old TLS versions

```
$ nmap  
--script ssl-enum-ciphers  
-p 443 wizardzines.com  
list all scripts with:  
$ nmap --script-help '*'
```

netcat

nc -l PORT

lets you create TCP (or UDP) connections from the command line



start a server! this listens on PORT and prints everything received (to send UDP use -u)



make HTTP requests by hand

```
printf 'GET / HTTP/\nHost: example.com\r\n\r\n'| nc example.com 80
```

I hand wrote this HTTP request for you!

type in any weird HTTP request you want! :)

send files

want to send a 100 GB file to someone on the same wifi network? easy!

```
receiver: nc -1 8080 > file
sender: cat file | nc YOUR_IP 8080
```

I ❤️ this trick!
It works even if you're disconnected from the internet!

nc IP PORT

be a client! opens a TCP connection to IP:PORT (to send UDP use -u)



SSH

★ port forwarding ★

```
ssh user@host.com -NfL 3333:localhost:8888
```

↑ local port
remote port
Let's you view a remote server that's not on the internet in your browser.

17

ssh-copy-id

This script installs your SSH key on a host (over ssh)

```
$ ssh-copy-id user@host
```

(puts it in .ssh/authorized_keys etc)
installing a SSH key is surprisingly finicky so this script is helpful!

just run 1 command

```
$ ssh user@host uname -a
```

runs this command & exits

~ . Enter ~. closes the SSH connection. Useful if it's hanging!

ssh-agent

remembers your SSH key passphrase so you don't have to keep typing it

ssh alternative: keeps the connection open if you disconnect + reconnect later

.ssh/config
Lets you set, per host:
- username to use
- SSH key to use
- an alias!
so you can type \$ ssh ALIAS instead of ssh user@verylongdomain.com

miscellaneous networking tools

16

stunnel make a SSL proxy for an insecure server	rsync sync files over SSH or locally	whos is this domain registered?	zenmap GUI for nmap	sysctl configure Linux kernel's network stack
hping3 make any TCP packet	lsof what ports are being used?	ipcalc easily see what 13.21.2.3/25 means	p0f identify OS of hosts connecting to you	iperf benchmarking tools
wget download files	httpie like curl but friendlier	Python 3 -m http.server serve files from a directory	openvpn wireguard VPNs	links a browser in your terminal
aria2c a fancier wget	iftop/nethogs/ ntop/iptraf/ nload see what's using bandwidth	nftables new version of iptables	tcpflow capture and assemble TCP streams	telnet can help debug text network protocols

socat

q	order doesn't matter socat THING1 THING2 is the same as socat THING2 THING1	-V write all transferred data to stderr useful for debugging!
so	socat supports tcp sockets unix domain sockets SSL sockets pipes processes files UDP sockets ... and MORE!	so proxy from local HTTP port to remote server socat TCP-LISTEN:1337 TCP:domain.com:80
the basic syntax: socat THING1 THING2	so expose a unix domain socket on port 1337	so socat TCP-CONNECT: /path UNIX-CONNECT:

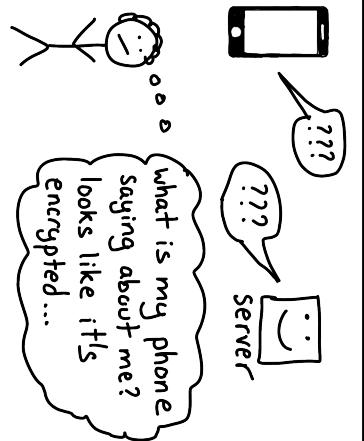
tcpdump

tcpdump lets you view network packets being sent + received

 it's not the easiest to use but it's usually installed ☺

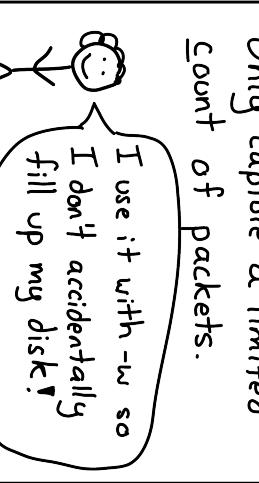
-w file.pcap

Write packets to a file for later analysis with tcpdump / tshark / wireshark / another tool
pcap is for "packet capture"



-n
don't try to resolve IP addresses / ports to DNS / port names. makes it run faster.

-A
print packet contents, not just headers. Nice if you want to quickly see what a few packets contain.

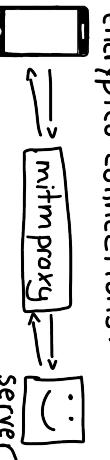
-c 100000
Only capture a limited count of packets.
 I often use "-i any" to make sure I'm not missing any packets!

mitmproxy

15

how you use it

- ① install mitmproxy root CA on your laptop / phone
- ② run mitmproxy web UI on computer
- ③ tell the program / phone to proxy through mitmproxy



mitmproxy can proxy connections from your laptop / phone and let you see the contents. It even works with encrypted connections!



some apps pin a cert makes mitmproxy not work, look up "trust killer" to get around that

other similar tools (not all free though)

- charles proxy
- burp suite
- fiddler

how it works

OpenSSL

`openssl` is a tool for doing *SSL things* aka TLS

- create CSRs
- inspect certificates
- sign certificates

It uses the OpenSSL library (or Libressl)

\$ `openssl x509 -in FILE.crt -noout -text`
this works for files ending in .crt or . pem! Try it out: You probably have certs in /usr/share/ca-certificates

please upload a CSR certificate authority **WHAT!**

to get a SSL cert for your website, you need to make a file called a "certificate signing request".

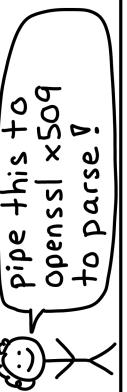
make a CSR

\$ `openssl req -new -sha256 -key FILE.key -out FILE.csr`

make one of these with
\$ `openssl genrsa`

look at a website's certificate

\$ `openssl s_client -showcerts -connect google.com:443`



md5 / sha1 / sha256 / sha512

Not quite SSL but useful:
\$ `openssl md5 FILE computes the md5sum of FILE. Same for other digests`

\$ `openssl list -digests -commands shows all supported digests.`

BPF cheat sheet

Berkeley Packet Filter
is a small language you can use to filter which packets tcpdump and ngrep capture.

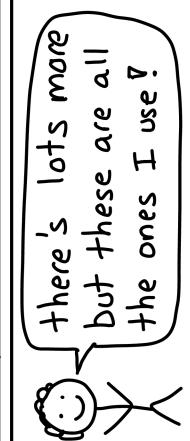
Use it like this:

```
$ tcpdump [your bpf here]
$ ngrep [your bpf here]
```

and / or / not

host 127.0.0.1 and port 80
udp and port 53
(port 53 or port 99) and not host 127.0.0.1

port
src port 53
port 80
again, same as "src or dst port"
less / greater
Packet length!
less 80
greater 200

tcp / udp / icmp
IPv4>ip / ip6
only show packets using that protocol

PROTOCOL [INDEX]
filter based on a specific byte in a packet
IP packets with options:
ip[0] & 0XF == 5
DNS SERVFAIL responses:
udp[11] & 0XF > 0
SYN packets:
tcp[tcpflags] == tcp-syn

tshark

Wireshark is an amazing graphical packet analysis tool

tshark is the command line version of Wireshark
it can do **much more things** than tcpdump

-Y
filter which packets are captured
tshark -Y
'http.request.method' == "GET"
↑ uses Wireshark's SUPER POWERFUL filter language

-T FORMAT
Output format. My favourites:
for these you can specify which fields you want with -e
* json
* fields: csv/tsv

* text: default summary

-e
Which fields to output. Ex:
\$ tshark -T fields
-e http.request.method
-e http.request.uri
-e ip.dst ↗ supports WAY more protocols than HTTP
GET /foo 92.183.216.34
POST /bar 10.23.38.132

-r file.pcap
analyze packets from a file instead of the network
-W ↗ same as tcpdump's
Write captured packets to a file. If -w file.pcap has permission issues, try: tshark -w - > file.pcap

ngrep

13

like grep for your network!

\$ sudo ngrep GET
will find every plaintext HTTP GET request

ngrep syntax

\$ ngrep [options] [regular expression] [BPF filter]

what to search
↓
packets for
regular expression
[BPF filter]

↑
same format as tcpdump uses!



-d
is for device
which network interface to use. same as tcpdump's (-i) (try '-d any'!)

-W byline
prints line breaks as line breaks, not "\n". Nice when looking at HTTP requests

-T file.pcap
-O file.pcap
read/write packets from/to a pcap file